Course Info

The purpose of this course is to teach the design of operating systems and operating systems concepts that appear in other computer systems. Topics we will cover include concepts of operating systems, systems programming, networked and distributed systems, and storage systems, including multiple-program systems (processes, interprocess communication, and synchronization), memory allocation (segmentation, paging), resource allocation and scheduling, file systems, basic networking (sockets, layering, APIs, reliability), transactions, security, and privacy.

We will be using the Pintos educational operating system for all three projects.

Homeworks and projects will all be submitted and autograded via GitHub. Individuals and groups will have GitHub repositories. Intermediate pushes will help the staff see how students are progressing.

Project teams must have 4 people. If the class does not divide evenly into groups of 4, then a few select groups may have a size of 3.

We are using the textbook, “Operating Systems: Principles and Practice” by Anderson and Dahlin (A&D). The reading from this textbook is **required**. There will be some additional readings that are not from this textbook that are freely available online.

Prerequisites

CS 61A, CS 61B, CS 61C, and CS 70. This means that you understand:

- Data structures: arrays, linked lists, binary trees, and hashing
- Assembly language programming
- The C programming language
- Debugging C using GDB
- CPU caches and memory hierarchy
- Virtual memory as covered in CS 61C
- CPU pipelines and basic digital logic design
- Basic knowledge of random variables and probability distributions as covered in CS 70

The TAs will spend a small amount of time reviewing some of the material you are less likely to remember. We will assume that you either know the material that is supposed to be covered in those courses, or that you are willing to learn the material as necessary. We will not spend time in lecture covering any of this material. **Perhaps more important than formal prequisites, however, is experience and maturity with debugging large programs, designing and implementing useful abstractions, and computational problem solving in general. This class will exercise your skills in these areas.**

If you feel that you would benefit from additional review of these prerequisites, we recommend using the following resources:

1. For a good review of probability as you’ll need it in this class, read through CS 70 Lecture Notes 15 (https://inst.eecs.berkeley.edu/~cs70/sp14/notes/n15.pdf), 16 (https://inst.eecs.berkeley.edu/~cs70/sp14/notes/n16.pdf), 17

(https://inst.eecs.berkeley.edu/~cs70/sp14/notes/n17.pdf), 18 (https://inst.eecs.berkeley.edu/~cs70/sp14/notes/n18.pdf), and 19 (https://inst.eecs.berkeley.edu/~cs70/sp14/notes/n19.pdf).

2. Hennessy, John L. and Patterson, David A. *Computer Architecture: A Quantitative Approach*. 5th edition. **Although the main content of this book goes far more in-depth into computer architecture than you are expected to know for this class, the appendices provide an excellent overview of some of the prerequisites. Focus specifically on the sections below:**

- 2.1 and B.1 - B.4 (you can skip the sixth optimization in B.3) review caching, virtual memory, and memory hierarchies.
- C.1 - C.2 review CPU pipelines at the level of CS 61C.
- This book is available at the Engineering Library on campus: check OskiCat (http://oskicat.berkeley.edu/record=b18534483~S1).
- Other editions (some older, some newer) are also available at the Engineering Library.

Enrollment

All decisions concerning appeals for enrolling in the course are made by CS departmental staff; the course staff have no say in the matter. If you have questions or concerns, feel free to contact the departmental staff.

Grading

At the end of the term, each student is assigned a score out of 100 points. It is determined as follows:

- 36% Exams
- 36% Projects
- 18% Homeworks
- 10% Participation

It is important that you attend discussion sections and get to know your TA. Section and the TAs will provide essential specifics on hands-on aspects of the course, including tools, techniques and concepts. In the past, TAs have bumped borderline grade cases up or down based on participation.

Discussion attendance and design review attendance is mandatory and your cameras must be ON when in online discussion sections and design reviews. This will count towards your participation grade.

Once students' raw scores are computed as described above, final grades are assigned using a curved scale.

Exams

3 midterms are scheduled, with no final exam. Each exam will be cumulative and cover material seen in the course so far. Please consult the class schedule for the dates and times of these exams. If you have a conflict, let us know, and we will schedule a makeup exam. Exams will cover material from lecture, sections, the readings, homework assignments, and projects. **You are likely to do poorly on the exams if you do not do your share of work on the project.**

Exams will be delivered in a remote format. Our exams will be closed-book. You *may* be allowed some number of handwritten cheat sheets, which we will specify prior to each exam. You are allowed to write these on a tablet and print them out. **You may *not* collaborate with other people, either in person or online, during the midterm exams.**

The weight of each exam is:

- 12% Midterm 1
- 12% Midterm 2
- 12% Midterm 3

Homework

We see homework mostly as a chance for exercise. Solutions will not be made available online, but you may come to Office Hours to see them 2 days after all possible slip days on the assignment have expired.

Project and Sections

The projects in this course provide a deep experience with operating system and distributed system design and implementation. We have tried hard to keep the workload manageable and to focus on learning concepts, rather than busy work. The project experience is essential to the course.

If you plan to do software or hardware development after graduation, you will almost certainly need to know how to work in a group. Recent CS grads almost all say that the ability to work in groups was the single most important thing they wished they had learned at Berkeley. Hence, for this project, you will need to form into groups of 4 people; the assignments will be the same no matter what size group you have. *We will not permit anyone to do the projects in a group smaller than 3, and we will only permit a few groups to have a size of three, if the class is not evenly divisible by 4.*. In order to ensure everyone in the group does their fair share of the work, we will ask each of you to turn in assessments of the relative contributions of your project partners. These assessments are important, and they can have a *substantial* impact on your grade if you do not participate.

TAs will grade all parts of your project. The TAs have been instructed to grade in part on design. In other words, it is **not** enough to get a working solution; you must implement the solution in a clean way that would simplify making further enhancements. (Several employers in the area have said that many of our graduates don't know how to program well — it will really benefit you in the long run to work on your software engineering skills.)

For each project, you will turn in an initial design document and a TA will meet with your group for an oral presentation of your design at a **design review**. These reviews serve several purposes:

1. To encourage you to get an early start on the assignments (a key to success in this course).
2. To catch design errors early, **before** you spend a lot of time debugging.
3. To give you an opportunity to explain and defend your approach (this is an important skill to learn as engineers).
4. To provide an opportunity to assess your understanding of the project.

Following the review, you will turn in the actual project code.

Piazza Policy

See the etiquette post (https://piazza.com/class/kjffms7injc3r5?cid=15) on Piazza.

OH Policy

When submitting a ticket to the OH queue, you must use the ticket template provided on the queue (https://oh.cs162.org/).

Students will be required to copy and paste the template into the "Issue" input box and fill out the template (deleting sections where necessary). If students don't have a templated ticket, the ticket may be ignored and you will not receive help until the formatting has been fixed.

Late Policy

All assignments are due at 11:59:59 PM Pacific Time on the day listed on the schedule.

Your group gets **7** project "slip days". You will not need to use project slip days twice for code and final report deadlines. Instead, we will apply the maximum number of slip days used for *either* submission for each project. However, project slip days can **not** be used for design docs. After the project slip days are used up, no credit will be given for late project submissions.

There are **5** homework "slip days". After the homework slip days are used up, no credit will be given for late homework submissions.

Slip days can only be used in whole numbers. Although no credit will be given for submissions after slip days have expired, we *may* be lenient in special circumstances, and grant manual extensions on assignments at our discretion.

Collaboration Policy

Note: Projects are a shared responsiblity and all project members will incur penalties for cheating. You are accountable for the actions of your teammates.
We encourage you to ask other students ***in this semester's course*** about the concepts, algorithms, or approaches needed to do the projects and assignments; both giving and taking advice will help you to learn. However, what you turn in must be your own, or for projects, your group's own work; copying other people's code, solution sets, or from any other sources, including online sources, is strictly prohibited. The project assignments must be the work of the students turning them in. Wherever you have benefited from the work of others, you should credit it properly in your code and/or writeup. Note: Only projects are group assignments. Homeworks are individual and you should not look at or copy other code, even that of your project groupmates.

Examples of acceptable collaboration between students in different project groups ***in this semester's course***:

- Explaining a concept to another student, or asking another student to explain a concept to you
- Discussing various algorithms or approaches for project components
- Discussing testing strategies and approaches
- Searching online for algorithms or implementations of generic (non-project-specific) abstractions (e.g., searching for various implementations of a hash table)
- Helping another student with debugging approaches (note that it is **not** acceptable to give that student code solutions or even look at their code, as described below)

Examples of unacceptable collaboration:

- Looking at code from a different group's project or another student's homework — this includes looking at online code from prior semesters or other institutions
- Using code from a different group's project or another student's homework — this includes incorporating online code from prior semesters or other institutions
- Looking at or using specific test case instances from a different group's project or another student's homework — this includes incorporating online code from prior semesters or other institutions
- Searching online for specific implementations of project abstractions or functions

We use an automated system for detecting cheating: it performs a pairwise comparison of all assignment submissions with all others for this class, prior semester classes, and various online repositories. The system reports any suspicious similarities. The course staff will manually review any such similarities. Moreover, we have

a dedicated staff of readers who will manually review code, and note similarities between algorithms and designs used in other students solutions as well as any solutions available online. Do NOT try to fool our automatic cheating detection system, because you WILL be caught by our human readers.

If two assignments are determined to be obviously very similar (i.e., we believe that they were done together or one was copied from the other), then all the students involved the incident, both the copy-er and the copy-ee, will receive a penalty. The minimum penalty is to receive a negative score on the assignment. More serious cases of academic dishonesty, such as repeat offences or cheating on exams, will incur a greater penalty at the instructor's discretion, such as receiving an F in the class. In addition, for every instance, a letter to the Center of Student Conduct will be attached to your permanent record, and a copy will be placed in the CS division office.

Note that you are responsible for not leaving copies of your assignments lying around and for protecting your files — do not use public unprotected source code repositories to store your code. You must set up your files and directories so that they are protected from others reading them.

Any mechanisms used in attempt to "hardcode" solutions to homeworks or projects will be punishable as cheating cases (leading to negative scores on assignments, an F in the class, a letter to the CSC, etc.) We will have hidden test cases on which any hardcoded solutions will fail, as well as readers who will manually inspect your code solution for any such instances of cheating.

Course Material Policy

Under no circumstances are students permitted to upload course materials online or distribute these materials to anyone not enrolled in the course. This policy will be enforced **during** and **after** the course.

This policy applies to any course materials that are not already publicly available, including but not limited to:

- Exam Questions and Solutions
- Design Documents and Final Reports
- Homework and Project Solutions

Please see the Collaboration Policy for information on sharing course materials among enrolled students.

Reading

Required

Operating Systems: Principles and Practice (2nd Edition) (http://ospp.cs.washington.edu/)

Recommended

Operating Systems: Three Easy Pieces (http://pages.cs.wisc.edu/~remzi/OSTEP/)

Supplemental

Linux Kernel Development (3rd Edition) (http://www.amazon.com/Linux-Kernel-Development-3rd-Edition/dp/0672329468)